Non-intrusive model reduction of shift-equivariant systems via data-driven projection and reduced dynamics

Yu Shuai

October 1, 2025

2025-10-01

Non-intrusive model reduction of shift-equivariant systems via data-driven projection and reduced dynamics

Non-intrusive model reduction of shift-equivariant systems via data-driven projection and reduced dynamics

Yu Shuai

October 1, 2025

Contents

Contents

Method

Results

Contents

Contents
Method
Symmetry-reduced non-intrusive trajectory-based ROM (SR-NiTROM)
Results

Formulation of the SR-NiTROM

- Consider a shift-equivariant FOM with traveling solutions:

$$u_t = f(u), \qquad u(x,t) = \hat{u}(x - c(t), t). \tag{1}$$

Formulation of the SR-NiTROM

- Consider a shift-equivariant FOM with traveling solutions:

$$u_t = f(u), \qquad u(x,t) = \hat{u}(x - c(t), t). \tag{1}$$

Formulation of the SR-NiTROM

- Consider a shift-equivariant FOM with traveling solutions:

$$u_t = f(u), \qquad u(x,t) = \hat{u}(x - c(t), t). \tag{1}$$

- Suppose we have a collection of training snapshots $\{u(t_m)\}_{m=0}^{N_t-1}$, $u(t) \in \mathbb{R}^N$.

Formulation of the SR-NiTROM
Consider a shift-equivariant FOM with traveling solutions:
$u_t = f(u), \qquad u(x,t) = \hat{u}(x - c(t), t). \quad (1)$
Suppose we have a collection of training snapshots $\{u(t_m)\}_{m=0}^{N_t-1}$, $u(t) \in \mathbb{R}^N$.

Formulation of the SR-NiTROM

- Consider a shift-equivariant FOM with traveling solutions:

$$u_t = f(u), \qquad u(x,t) = \hat{u}(x - c(t), t). \tag{1}$$

- Suppose we have a collection of training snapshots $\{u(t_m)\}_{m=0}^{N_t-1}$, $u(t) \in \mathbb{R}^N$.
- We seek to find $\Phi, \Psi \in \mathbb{R}^{n\times r}$, such that $\Phi(\Psi^\top\Phi)^{-1}\Psi^\top$ is a projection.

Formulation of the SR-NiTROM
- Consider a shift-equivariant FOM with traveling solutions:
$$u_t = f(u), \qquad u(x,t) = \hat{u}(x - c(t), t). \tag{1}$$
- Suppose we have a collection of training snapshots $\{u(t_m)\}_{m=0}^{N_t-1}$, $u(t) \in \mathbb{R}^N$.
- We seek to find $\Phi, \Psi \in \mathbb{R}^{n\times r}$, such that $\Phi(\Psi^\top\Phi)^{-1}\Psi^\top$ is a projection.

Formulation of the SR-NiTROM

- Consider a shift-equivariant FOM with traveling solutions:

$$u_t = f(u), \qquad u(x,t) = \hat{u}(x - c(t), t). \tag{1}$$

- Suppose we have a collection of training snapshots $\{u(t_m)\}_{m=0}^{N_t-1}$, $u(t) \in \mathbb{R}^N$.
- We seek to find $\Phi, \Psi \in \mathbb{R}^{n \times r}$, such that $\Phi(\Psi^\top \Phi)^{-1}\Psi^\top$ is a projection.
- From this projection operator, we can encode the FOM state with a low-dim representation $a(t) \in \mathbb{R}^r$:

$$\begin{aligned} a &= \Psi^\top u \\ \hat{u}_r &= \Phi(\Psi^\top \Phi)^{-1} a. \end{aligned} \tag{2}$$

Formulation of the SR-NiTROM

- Consider a shift-equivariant FOM with traveling solutions:

$$u_t = f(u), \qquad u(x,t) = \hat{u}(x - c(t), t). \tag{1}$$

- Suppose we have a collection of training snapshots $\{u(t_m)\}_{m=0}^{N_t-1}$, $u(t) \in \mathbb{R}^N$.
- We seek to find $\Phi, \Psi \in \mathbb{R}^{n \times r}$, such that $\Phi(\Psi^\top \Phi)^{-1}\Psi^\top$ is a projection.
- From this projection operator, we can encode the FOM state with a low-dim representation $a(t) \in \mathbb{R}^r$:

$$\begin{aligned} a &= \Psi^\top u \\ \hat{u}_r &= \Phi(\Psi^\top \Phi)^{-1} a. \end{aligned} \tag{2}$$

Formulation of the SR-NiTROM

- Consider a shift-equivariant FOM with traveling solutions:

$$u_t = f(u), \qquad u(x,t) = \hat{u}(x - c(t), t). \tag{1}$$

- Suppose we have a collection of training snapshots $\{u(t_m)\}_{m=0}^{N_t-1}$, $u(t) \in \mathbb{R}^N$.
- We seek to find $\Phi, \Psi \in \mathbb{R}^{n\times r}$, such that $\Phi(\Psi^\top \Phi)^{-1}\Psi^\top$ is a projection.
- From this projection operator, we can encode the FOM state with a low-dim representation $a(t) \in \mathbb{R}^r$:

$$\begin{aligned} a &= \Psi^\top u \\ \hat{u}_r &= \Phi(\Psi^\top \Phi)^{-1} a. \end{aligned} \tag{2}$$

- The ROM dynamics is given in a symmetry-reduced form:

$$\dot{a}_i = A_{ij}a_j + B_{ijk}a_j a_k + \dot{c}M_{ij}a_j \tag{3a}$$

$$\dot{c} = -\frac{p_i a_i + Q_{ij}a_i a_j}{s_i a_i} \tag{3b}$$

$$M = \Psi^\top \partial_x \Phi(\Psi^\top \Phi)^{-1}, \quad s = \langle \partial_x \Phi(\Psi^\top \Phi)^{-1}, \partial_x u_0 \rangle \tag{3c}$$

Formulation of the SR-NiTROM

- Consider a shift-equivariant FOM with traveling solutions:
$$u_t = f(u), \qquad u(x,t) = \hat{u}(x - c(t), t). \tag{1}$$
- Suppose we have a collection of training snapshots $\{u(t_m)\}_{m=0}^{N_t-1}$, $u(t) \in \mathbb{R}^N$.
- We seek to find $\Phi, \Psi \in \mathbb{R}^{n\times r}$, such that $\Phi(\Psi^\top \Phi)^{-1}\Psi^\top$ is a projection.
- From this projection operator, we can encode the FOM state with a low-dim representation $a(t) \in \mathbb{R}^r$:
$$\begin{aligned} a &= \Psi^\top u \\ \hat{u}_r &= \Phi(\Psi^\top \Phi)^{-1} a. \end{aligned} \tag{2}$$
- The ROM dynamics is given in a symmetry-reduced form:
$$\dot{a}_i = A_{ij}a_j + B_{ijk}a_j a_k + \dot{c}M_{ij}a_j \tag{3a}$$
$$\dot{c} = -\frac{p_i a_i + Q_{ij}a_i a_j}{s_i a_i} \tag{3b}$$
$$M = \Psi^\top \partial_x \Phi(\Psi^\top \Phi)^{-1}, \quad s = \langle \partial_x \Phi(\Psi^\top \Phi)^{-1}, \partial_x u_0 \rangle \tag{3c}$$

The optimization problem of SR-NiTROM

- The trajectory-based objective function:

$$J = \sum_{m=0}^{N_t-1} \|\hat{u}_r(t_m) - \hat{u}(t_m)\|^2 + \beta(c_r(t_m) - c(t_m))^2. \tag{4}$$

2025-10-01

The optimization problem of SR-NiTROM

- The trajectory-based objective function:

$$J = \sum_{m=0}^{N_t-1} \|\hat{u}_r(t_m) - \hat{u}(t_m)\|^2 + \beta(c_r(t_m) - c(t_m))^2. \tag{4}$$

The optimization problem of SR-NiTROM

- The trajectory-based objective function:

$$J = \sum_{m=0}^{N_t-1} \|\hat{u}_r(t_m) - \hat{u}(t_m)\|^2 + \beta(c_r(t_m) - c(t_m))^2. \tag{4}$$

  - $\beta = \gamma \sum_m \|\hat{u}(t_m)\|^2 / \sum_m (c(t_0) - c(t_m))^2$: relative weights.

The optimization problem of SR-NiTROM
The trajectory-based objective function:
$$J = \sum_{m=0}^{N_t-1} \|\hat{u}_r(t_m) - \hat{u}(t_m)\|^2 + \beta(c_r(t_m) - c(t_m))^2. \tag{4}$$
$\beta = \gamma \sum_m \|\hat{u}(t_m)\|^2 / \sum_m (c(t_0) - c(t_m))^2$: relative weights.

The optimization problem of SR-NiTROM

- The trajectory-based objective function:

$$J = \sum_{m=0}^{N_t-1} \|\hat{u}_r(t_m) - \hat{u}(t_m)\|^2 + \beta(c_r(t_m) - c(t_m))^2. \tag{4}$$

  - $\beta = \gamma \sum_m \|\hat{u}(t_m)\|^2 / \sum_m (c(t_0) - c(t_m))^2$: relative weights.
  - γ: hyperparameter.

The optimization problem of SR-NiTROM
The trajectory-based objective function:
$$J = \sum_{m=0}^{N_t-1} \|\hat{u}_r(t_m) - \hat{u}(t_m)\|^2 + \beta(c_r(t_m) - c(t_m))^2. \tag{4}$$
$\beta = \gamma \sum_m \|\hat{u}(t_m)\|^2 / \sum_m (c(t_0) - c(t_m))^2$: relative weights.
γ: hyperparameter.

The optimization problem of SR-NiTROM

- The trajectory-based objective function:

$$J = \sum_{m=0}^{N_t-1} \left\|\hat{u}_r(t_m) - \hat{u}(t_m)\right\|^2 + \beta(c_r(t_m) - c(t_m))^2. \tag{4}$$

  - $\beta = \gamma \sum_m \|\hat{u}(t_m)\|^2 / \sum_m (c(t_0) - c(t_m))^2$: relative weights.
  - γ: hyperparameter.
- The unconstrained Lagrangian with multipliers:

$$L = \sum_{m=0}^{N_t-1} \Bigg(\left\|\hat{u}_r(t_m) - \hat{u}(t_m)\right\|^2 + \beta(c_r(t_m) - c(t_m))$$

$$+ \int_{t_0}^{t_m} \lambda_m^\top(\dot{a} - Aa - B(a,a) - \dot{c}Ma)\mathrm{d}t \tag{5}$$

$$+ \int_{t_0}^{t_m} \mu_m(\dot{c} + \frac{p_i a_i + Q_{ij} a_i a_j}{s_i a_i})\mathrm{d}t \tag{6}$$

$$+ \lambda_m(t_0)(a(t_0) - \Psi^\top \hat{u}(t_0)) \Bigg), \quad \lambda_m \in \mathbb{R}^r, \mu_m \in \mathbb{R}. \tag{7}$$

The optimization problem of SR-NiTROM
The trajectory-based objective function: (4)
$\beta = \gamma \sum_m \|\hat{u}(t_m)\|^2 / \sum_m (c(t_0) - c(t_m))^2$: relative weights.
γ: hyperparameter.
The unconstrained Lagrangian with multipliers: (5) (6) (7)

Contents

Contents
Method
Results
Single trajectory: SR-NiTROM vs SR-Galerkin

Numerical details

- FOM: Kuramoto-Sivashinsky equation

$$u_t = -uu_x - u_{xx} - \nu u_{xxxx}, \quad x \in [0, 2\pi]. \tag{8}$$

Numerical details
FOM: Kuramoto-Sivashinsky equation
$u_t = -uu_x - u_{xx} - \nu u_{xxxx}, \quad x \in [0, 2\pi].$ (8)

Numerical details

- FOM: Kuramoto-Sivashinsky equation

$$u_t = -uu_x - u_{xx} - \nu u_{xxxx}, \quad x \in [0, 2\pi]. \tag{8}$$

  - $\nu = 4/87$ for traveling-wave patterns.

Numerical details
FOM: Kuramoto-Sivashinsky equation
$$u_t = -uu_x - u_{xx} - \nu u_{xxxx}, \quad x \in [0, 2\pi]. \tag{8}$$
$\nu = 4/87$ for traveling-wave patterns.

Numerical details

- FOM: Kuramoto-Sivashinsky equation

$$u_t = -uu_x - u_{xx} - \nu u_{xxxx}, \quad x \in [0, 2\pi]. \tag{8}$$

  - $\nu = 4/87$ for traveling-wave patterns.
  - Periodic BCs, $N = 40$ Fourier modes, $\Delta t = 10^{-3}$.

Numerical details
FOM: Kuramoto-Sivashinsky equation
$u_t = -uu_x - u_{xx} - \nu u_{xxxx}, \quad x \in [0, 2\pi].$ (8)
$\nu = 4/87$ for traveling-wave patterns.
Periodic BCs, $N = 40$ Fourier modes, $\Delta t = 10^{-3}$.

Numerical details

- FOM: Kuramoto-Sivashinsky equation

$$u_t = -uu_x - u_{xx} - \nu u_{xxxx}, \quad x \in [0, 2\pi]. \tag{8}$$

  - $\nu = 4/87$ for traveling-wave patterns.
  - Periodic BCs, $N = 40$ Fourier modes, $\Delta t = 10^{-3}$.
  - Sample interval: 10 timesteps between 2 adjacent snapshots.

Numerical details

- FOM: Kuramoto-Sivashinsky equation

$$u_t = -uu_x - u_{xx} - \nu u_{xxxx}, \quad x \in [0, 2\pi]. \tag{8}$$

  - $\nu = 4/87$ for traveling-wave patterns.
  - Periodic BCs, $N = 40$ Fourier modes, $\Delta t = 10^{-3}$.
  - Sample interval: 10 timesteps between 2 adjacent snapshots.

Numerical details

- FOM: Kuramoto-Sivashinsky equation

$$u_t = -uu_x - u_{xx} - \nu u_{xxxx}, \quad x \in [0, 2\pi]. \tag{8}$$

  - $\nu = 4/87$ for traveling-wave patterns.
  - Periodic BCs, $N = 40$ Fourier modes, $\Delta t = 10^{-3}$.
  - Sample interval: 10 timesteps between 2 adjacent snapshots.
- Optimization of the SR-NiTROM:

Numerical details
- FOM: Kuramoto-Sivashinsky equation
$$u_t = -uu_x - u_{xx} - \nu u_{xxxx}, \quad x \in [0, 2\pi]. \tag{8}$$
  - $\nu = 4/87$ for traveling-wave patterns.
  - Periodic BCs, $N = 40$ Fourier modes, $\Delta t = 10^{-3}$.
  - Sample interval: 10 timesteps between 2 adjacent snapshots.
- Optimization of the SR-NiTROM:

- FOM: Kuramoto-Sivashinsky equation

$$u_t = -uu_x - u_{xx} - \nu u_{xxxx}, \quad x \in [0, 2\pi]. \tag{8}$$

  - $\nu = 4/87$ for traveling-wave patterns.
  - Periodic BCs, $N = 40$ Fourier modes, $\Delta t = 10^{-3}$.
  - Sample interval: 10 timesteps between 2 adjacent snapshots.
- Optimization of the SR-NiTROM:
  - coordinate-descent method conjugate gradient optimizer for each subproblems.

Numerical details

- FOM: Kuramoto-Sivashinsky equation

$$u_t = -uu_x - u_{xx} - \nu u_{xxxx}, \quad x \in [0, 2\pi]. \tag{8}$$

  - $\nu = 4/87$ for traveling-wave patterns.
  - Periodic BCs, $N = 40$ Fourier modes, $\Delta t = 10^{-3}$.
  - Sample interval: 10 timesteps between 2 adjacent snapshots.
- Optimization of the SR-NiTROM:
  - coordinate-descent method conjugate gradient optimizer for each subproblems.

Numerical details

- FOM: Kuramoto-Sivashinsky equation

$$u_t = -uu_x - u_{xx} - \nu u_{xxxx}, \quad x \in [0, 2\pi]. \tag{8}$$

  - $\nu = 4/87$ for traveling-wave patterns.
  - Periodic BCs, $N = 40$ Fourier modes, $\Delta t = 10^{-3}$.
  - Sample interval: 10 timesteps between 2 adjacent snapshots.
- Optimization of the SR-NiTROM:
  - coordinate-descent method conjugate gradient optimizer for each subproblems.
  - 20 outer loops, 5 CG updates per outer loops.

Numerical details

- FOM: Kuramoto-Sivashinsky equation

$$u_t = -uu_x - u_{xx} - \nu u_{xxxx}, \quad x \in [0, 2\pi]. \tag{8}$$

  - $\nu = 4/87$ for traveling-wave patterns.
  - Periodic BCs, $N = 40$ Fourier modes, $\Delta t = 10^{-3}$.
  - Sample interval: 10 timesteps between 2 adjacent snapshots.
- Optimization of the SR-NiTROM:
  - coordinate-descent method conjugate gradient optimizer for each subproblems.
  - 20 outer loops, 5 CG updates per outer loops.

- FOM: Kuramoto-Sivashinsky equation

$$u_t = -uu_x - u_{xx} - \nu u_{xxxx}, \quad x \in [0, 2\pi]. \tag{8}$$

  - $\nu = 4/87$ for traveling-wave patterns.
  - Periodic BCs, $N = 40$ Fourier modes, $\Delta t = 10^{-3}$.
  - Sample interval: 10 timesteps between 2 adjacent snapshots.
- Optimization of the SR-NiTROM:
  - coordinate-descent method conjugate gradient optimizer for each subproblems.
  - 20 outer loops, 5 CG updates per outer loops.
  - Initial conditions: POD bases (capturing $>$99.5% energy) + Galerkin-projected tensors. (imitating the training result of the re-projected SR-OpInf ROM)

Numerical details

- FOM: Kuramoto-Sivashinsky equation

$$u_t = -uu_x - u_{xx} - \nu u_{xxxx}, \quad x \in [0, 2\pi]. \tag{8}$$

  - $\nu = 4/87$ for traveling-wave patterns.
  - Periodic BCs, $N = 40$ Fourier modes, $\Delta t = 10^{-3}$.
  - Sample interval: 10 timesteps between 2 adjacent snapshots.
- Optimization of the SR-NiTROM:
  - coordinate-descent method conjugate gradient optimizer for each subproblems.
  - 20 outer loops, 5 CG updates per outer loops.
  - Initial conditions: POD bases (capturing $>$99.5% energy) + Galerkin-projected tensors. (imitating the training result of the re-projected SR-OpInf ROM)

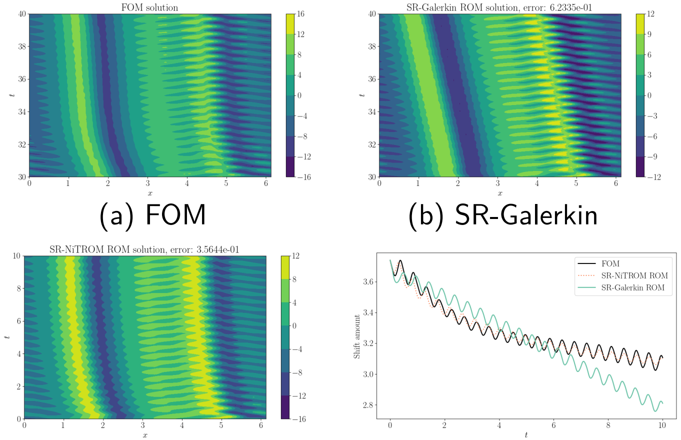
Results: single transient trajectory from t = 30 to t = 40

- Relative weight: 10.0, 4-dim ROM

Results: single transient trajectory from t = 30 to t = 40
Relative weight: 10.0, 4-dim ROM

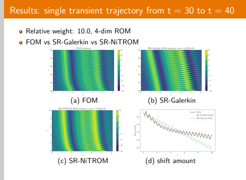
Results: single transient trajectory from t = 30 to t = 40

- Relative weight: 10.0, 4-dim ROM
- FOM vs SR-Galerkin vs SR-NiTROM



(a) FOM (b) SR-Galerkin

(c) SR-NiTROM (d) shift amount



Results: multiple trajectories from perturbed initial conditions

- 9 trajectories, 7-dim ROM

2025-10-01

Non-intrusive model reduction of shift-equivariant systems via data-driven projection and reduced dynamics
└─Results
  └─Single trajectory: SR-NiTROM vs SR-Galerkin

Results: multiple trajectories from perturbed initial conditions

- 9 trajectories, 7-dim ROM

Results: multiple trajectories from perturbed initial conditions

- 9 trajectories, 7-dim ROM
- Initial conditions: post-transient solution snapshot + perturbations $u(t = 80) + \{0, \sin(x), ..., \sin(4x), \cos(x), ..., \cos(4x)\}$.

Results: multiple trajectories from perturbed initial conditions

- 9 trajectories, 7-dim ROM
- Initial conditions: post-transient solution snapshot + perturbations $u(t = 80) + \{0, \sin(x), ..., \sin(4x), \cos(x), ..., \cos(4x)\}$.

Results: multiple trajectories from perturbed initial conditions

- 9 trajectories, 7-dim ROM
- Initial conditions: post-transient solution snapshot + perturbations $u(t = 80) + \{0, \sin(x), ..., \sin(4x), \cos(x), ..., \cos(4x)\}$.
- Strategy A: 20 outer training loops (10 on bases, 10 on tensors), 5 CG updates per outer loop

Results: multiple trajectories from perturbed initial conditions
- 9 trajectories, 7-dim ROM
- Initial conditions: post-transient solution snapshot + perturbations $u(t = 80) + \{0, \sin(x), ..., \sin(4x), \cos(x), ..., \cos(4x)\}$.
- Strategy A: 20 outer training loops (10 on bases, 10 on tensors), 5 CG updates per outer loop

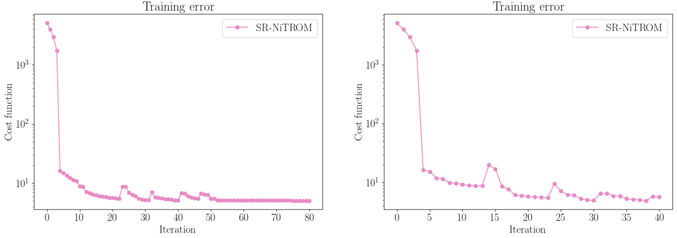
Results: multiple trajectories from perturbed initial conditions

- 9 trajectories, 7-dim ROM
- Initial conditions: post-transient solution snapshot + perturbations $u(t = 80) + \{0, \sin(x), ..., \sin(4x), \cos(x), ..., \cos(4x)\}$.
- Strategy A: 20 outer training loops (10 on bases, 10 on tensors), 5 CG updates per outer loop
- Strategy B: 10 outer training loops on tensors only with fixed POD bases

Results: multiple trajectories from perturbed initial conditions
- 9 trajectories, 7-dim ROM
- Initial conditions: post-transient solution snapshot + perturbations $u(t = 80) + \{0, \sin(x), ..., \sin(4x), \cos(x), ..., \cos(4x)\}$.
- Strategy A: 20 outer training loops (10 on bases, 10 on tensors), 5 CG updates per outer loop
- Strategy B: 10 outer training loops on tensors only with fixed POD bases

Results: multiple trajectories from perturbed initial conditions

- 9 trajectories, 7-dim ROM
- Initial conditions: post-transient solution snapshot + perturbations $u(t=80)+\{0, \sin(x), ..., \sin(4x), \cos(x), ..., \cos(4x)\}$.
- Strategy A: 20 outer training loops (10 on bases, 10 on tensors), 5 CG updates per outer loop
- Strategy B: 10 outer training loops on tensors only with fixed POD bases
- Training loss: not too much difference on the training set.



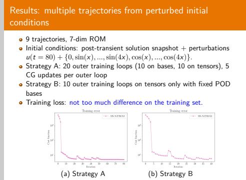
(a) Strategy A (b) Strategy B

2025-10-01

Non-intrusive model reduction of shift-equivariant systems via data-driven projection and reduced dynamics

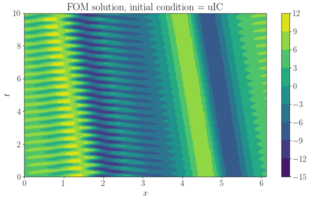
└Results

└Single trajectory: SR-NiTROM vs SR-Galerkin

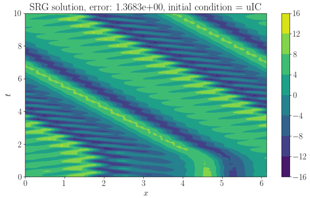


Results: multiple trajectories from perturbed initial conditions
9 trajectories, 7-dim ROM
Initial conditions: post-transient solution snapshot + perturbations $u(t=80)+\{0, \sin(x), ..., \sin(4x), \cos(x), ..., \cos(4x)\}$.
Strategy A: 20 outer training loops (10 on bases, 10 on tensors), 5 CG updates per outer loop
Strategy B: 10 outer training loops on tensors only with fixed POD bases
Training loss: not too much difference on the training set.
(a) Strategy A (b) Strategy B

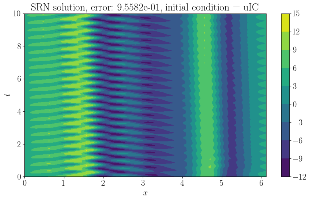
- Trajectory 1:



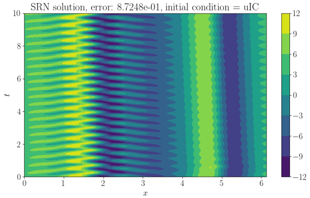
(a) FOM



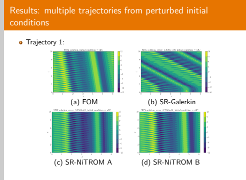
(b) SR-Galerkin



(c) SR-NiTROM A

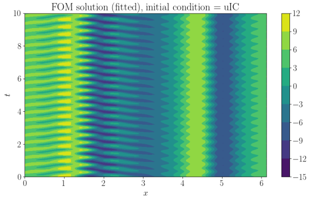


(d) SR-NiTROM B

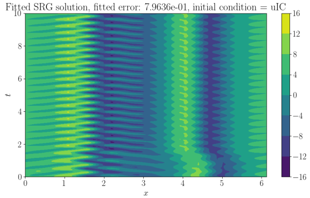


Results: multiple trajectories from perturbed initial conditions

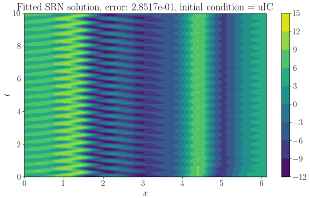
- Trajectory 1:



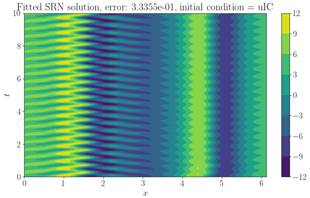
(a) FOM



(b) SR-Galerkin



(c) SR-NiTROM A



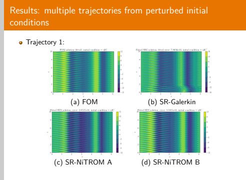
(d) SR-NiTROM B

2025-10-01

Non-intrusive model reduction of shift-equivariant systems via data-driven projection and reduced dynamics

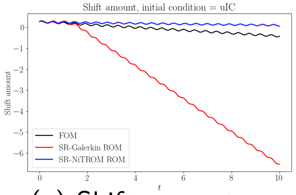
└Results

└Single trajectory: SR-NiTROM vs SR-Galerkin

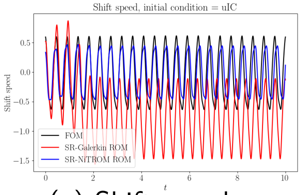


Results: multiple trajectories from perturbed initial conditions

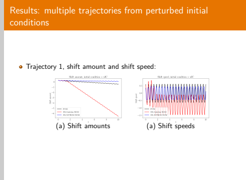
- Trajectory 1, shift amount and shift speed:



(a) Shift amounts



(a) Shift speeds



Conclusions:

- For the reconstruction of a single training trajectory including transient, SR-NiTROM outperforms SR-Galerkin ROM (and SR-OpInf of course).

Conclusions:
For the reconstruction of a single training trajectory including transient, SR-NiTROM outperforms SR-Galerkin ROM (and SR-OpInf of course).

Conclusions:

- For the reconstruction of a single training trajectory including transient, SR-NiTROM outperforms SR-Galerkin ROM (and SR-OpInf of course).
- For the reconstruction of multiple transient trajectories, we find that:

Conclusions:

- For the reconstruction of a single training trajectory including transient, SR-NiTROM outperforms SR-Galerkin ROM (and SR-OpInf of course).
- For the reconstruction of multiple transient trajectories, we find that:

Conclusions:

- For the reconstruction of a single training trajectory including transient, SR-NiTROM outperforms SR-Galerkin ROM (and SR-OpInf of course).
- For the reconstruction of multiple transient trajectories, we find that:
  - SR-NiTROM gives better approximation of template-aligned snapshots and shift amounts than the SR-Galerkin ROM.

Conclusions:

- For the reconstruction of a single training trajectory including transient, SR-NiTROM outperforms SR-Galerkin ROM (and SR-OpInf of course).
- For the reconstruction of multiple transient trajectories, we find that:
  - SR-NiTROM gives better approximation of template-aligned snapshots and shift amounts than the SR-Galerkin ROM.

Conclusions:

- For the reconstruction of a single training trajectory including transient, SR-NiTROM outperforms SR-Galerkin ROM (and SR-OpInf of course).
- For the reconstruction of multiple transient trajectories, we find that:
  - SR-NiTROM gives better approximation of template-aligned snapshots and shift amounts than the SR-Galerkin ROM.
  - It's better to optimize both the bases and the tensors to minimize our loss function.

Conclusions:

- For the reconstruction of a single training trajectory including transient, SR-NiTROM outperforms SR-Galerkin ROM (and SR-OpInf of course).
- For the reconstruction of multiple transient trajectories, we find that:
  - SR-NiTROM gives better approximation of template-aligned snapshots and shift amounts than the SR-Galerkin ROM.
  - It's better to optimize both the bases and the tensors to minimize our loss function.

Conclusions:

- For the reconstruction of a single training trajectory including transient, SR-NiTROM outperforms SR-Galerkin ROM (and SR-OpInf of course).
- For the reconstruction of multiple transient trajectories, we find that:
  - SR-NiTROM gives better approximation of template-aligned snapshots and shift amounts than the SR-Galerkin ROM.
  - It's better to optimize both the bases and the tensors to minimize our loss function.
  - However, SR-NiTROM with only trained tensors and POD bases attains the least reconstruction error of the raw snapshots.

Conclusions:
For the reconstruction of a single training trajectory including transient, SR-NiTROM outperforms SR-Galerkin ROM (and SR-OpInf of course).
For the reconstruction of multiple transient trajectories, we find that:
SR-NiTROM gives better approximation of template-aligned snapshots and shift amounts than the SR-Galerkin ROM.
It's better to optimize both the bases and the tensors to minimize our loss function.
However, SR-NiTROM with only trained tensors and POD bases attains the least reconstruction error of the raw snapshots.

Conclusions:

- For the reconstruction of a single training trajectory including transient, SR-NiTROM outperforms SR-Galerkin ROM (and SR-OpInf of course).
- For the reconstruction of multiple transient trajectories, we find that:
  - SR-NiTROM gives better approximation of template-aligned snapshots and shift amounts than the SR-Galerkin ROM.
  - It's better to optimize both the bases and the tensors to minimize our loss function.
  - However, SR-NiTROM with only trained tensors and POD bases attains the least reconstruction error of the raw snapshots.
    - Why: our loss function = error in aligned snapshots + error in shift amounts, not error in raw snapshots.

Conclusions:

- For the reconstruction of a single training trajectory including transient, SR-NiTROM outperforms SR-Galerkin ROM (and SR-OpInf of course).
- For the reconstruction of multiple transient trajectories, we find that:
  - SR-NiTROM gives better approximation of template-aligned snapshots and shift amounts than the SR-Galerkin ROM.
  - It's better to optimize both the bases and the tensors to minimize our loss function.
  - However, SR-NiTROM with only trained tensors and POD bases attains the least reconstruction error of the raw snapshots.
    - Why: our loss function = error in aligned snapshots + error in shift amounts, not error in raw snapshots.

Conclusions:

- For the reconstruction of a single training trajectory including transient, SR-NiTROM outperforms SR-Galerkin ROM (and SR-OpInf of course).
- For the reconstruction of multiple transient trajectories, we find that:
  - SR-NiTROM gives better approximation of template-aligned snapshots and shift amounts than the SR-Galerkin ROM.
  - It's better to optimize both the bases and the tensors to minimize our loss function.
  - However, SR-NiTROM with only trained tensors and POD bases attains the least reconstruction error of the raw snapshots.
    - Why: our loss function = error in aligned snapshots + error in shift amounts, not error in raw snapshots.
    - Trade-off: we may want to switch to error in raw snapshots, but then the optimizer doesn't know if the error comes from mismatch of aligned profiles or shift amounts.

Conclusions:

- For the reconstruction of a single training trajectory including transient, SR-NiTROM outperforms SR-Galerkin ROM (and SR-OpInf of course).
- For the reconstruction of multiple transient trajectories, we find that:
  - SR-NiTROM gives better approximation of template-aligned snapshots and shift amounts than the SR-Galerkin ROM.
  - It's better to optimize both the bases and the tensors to minimize our loss function.
  - However, SR-NiTROM with only trained tensors and POD bases attains the least reconstruction error of the raw snapshots.
    - Why: our loss function = error in aligned snapshots + error in shift amounts, not error in raw snapshots.
    - Trade-off: we may want to switch to error in raw snapshots, but then the optimizer doesn't know if the error comes from mismatch of aligned profiles or shift amounts.

Conclusions:

- For the reconstruction of a single training trajectory including transient, SR-NiTROM outperforms SR-Galerkin ROM (and SR-OpInf of course).
- For the reconstruction of multiple transient trajectories, we find that:
  - SR-NiTROM gives better approximation of template-aligned snapshots and shift amounts than the SR-Galerkin ROM.
  - It's better to optimize both the bases and the tensors to minimize our loss function.
  - However, SR-NiTROM with only trained tensors and POD bases attains the least reconstruction error of the raw snapshots.
    - Why: our loss function = error in aligned snapshots + error in shift amounts, not error in raw snapshots.
    - Trade-off: we may want to switch to error in raw snapshots, but then the optimizer doesn't know if the error comes from mismatch of aligned profiles or shift amounts.
    - This new loss function is reasonable since small shift mismatch can lead to large error in raw snapshots.

Conclusions:
- For the reconstruction of a single training trajectory including transient, SR-NiTROM outperforms SR-Galerkin ROM (and SR-OpInf of course).
- For the reconstruction of multiple transient trajectories, we find that:
  - SR-NiTROM gives better approximation of template-aligned snapshots and shift amounts than the SR-Galerkin ROM.
  - It's better to optimize both the bases and the tensors to minimize our loss function.
  - However, SR-NiTROM with only trained tensors and POD bases attains the least reconstruction error of the raw snapshots.
    - Why: our loss function = error in aligned snapshots + error in shift amounts, not error in raw snapshots.
    - Trade-off: we may want to switch to error in raw snapshots, but then the optimizer doesn't know if the error comes from mismatch of aligned profiles or shift amounts.
    - This new loss function is reasonable since small shift mismatch can lead to large error in raw snapshots.

Conclusions:

- For the reconstruction of a single training trajectory including transient, SR-NiTROM outperforms SR-Galerkin ROM (and SR-OpInf of course).
- For the reconstruction of multiple transient trajectories, we find that:
  - SR-NiTROM gives better approximation of template-aligned snapshots and shift amounts than the SR-Galerkin ROM.
  - It's better to optimize both the bases and the tensors to minimize our loss function.
  - However, SR-NiTROM with only trained tensors and POD bases attains the least reconstruction error of the raw snapshots.
    - Why: our loss function = error in aligned snapshots + error in shift amounts, not error in raw snapshots.
    - Trade-off: we may want to switch to error in raw snapshots, but then the optimizer doesn't know if the error comes from mismatch of aligned profiles or shift amounts.
    - This new loss function is reasonable since small shift mismatch can lead to large error in raw snapshots.
- To-dos: test our SR-NiTROM on unseen trajectories. Compute the obliqueness of projection.

2025-10-01

Conclusions:

- For the reconstruction of a single training trajectory including transient, SR-NiTROM outperforms SR-Galerkin ROM (and SR-OpInf of course).
- For the reconstruction of multiple transient trajectories, we find that:
  - SR-NiTROM gives better approximation of template-aligned snapshots and shift amounts than the SR-Galerkin ROM.
  - It's better to optimize both the bases and the tensors to minimize our loss function.
  - However, SR-NiTROM with only trained tensors and POD bases attains the least reconstruction error of the raw snapshots.
    - Why: our loss function = error in aligned snapshots + error in shift amounts, not error in raw snapshots.
    - Trade-off: we may want to switch to error in raw snapshots, but then the optimizer doesn't know if the error comes from mismatch of aligned profiles or shift amounts.
    - This new loss function is reasonable since small shift mismatch can lead to large error in raw snapshots.
- To-dos: test our SR-NiTROM on unseen trajectories. Compute the obliqueness of projection.